Amendments to the Specification:

Please amend the specification as follows:

Page 4, lines 2-23 to Page 6, lines 1-5:

The embodiments of the present invention fill these needs by providing an Extensible Markup Language (XML) based report generator. The XML based report generator of the embodiments of the present invention allows a test summary report to be generated from a test execution log file quickly, generally without manual intervention from a user, and consequently, reducing human induced errors. In one embodiment, a method for creating a test summary report is disclosed. Broadly speaking, a test is executed and the test results are generated in [[a]] an XML enabled format. The XML enabled test results are processed to create a test summary report.

In another embodiment, [[a]] an XML based report generator is disclosed. The XML based report generator includes a parser that processes a test execution log file to generate a well-formed XML test reports file. In addition, a logical parser is included that processes the well-formed XML test reports file to produce a logically arranged XML test reports file. The XML based report generator further includes a Hypertext Markup Language (HTML) converter parser that converts the logically arranged XML test reports file into [[a]] an HTML test summary file.

Another method for creating a test summary report is disclosed in a further embodiment of the present invention. The method includes executing a test application on a platform, where the test application is executed using a status utility having functions that generates XML code. The test results are generated in [[a]] an XML enabled format using the status utility, and are output to a test execution log file. The test execution log file is processed to generate a well-formed XML test reports file, which is then arranged to create a logically arranged XML test reports file. The logically arranged XML test reports file is then converted into [[a]] an HTML test summary report. Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

Page 7, lines 2 to 10:

An invention is disclosed for [[a]] an XML based report generator. The XML based report generator of the embodiments of the present invention allows a test summary report to be generated from a test execution log file quickly, generally without manual intervention from a user, and consequently, reducing human induced errors. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order not to unnecessarily obscure the present invention.

Page 8, lines 1-17 to page 10, lines 1-5:

Each test listing 208a-208b lists compile test results 210a-210b, execute test results 212a-212b, and a test result 214a-214b. The compile test results 210a-210b list information on the test compilation for the particular test 208a. For example, the compile test results 210a can list whether or not the test 208a compiled correctly, and if not, source code errors. The execute test results 212a-212b list information on the test execution of the particular test 208a. For example, the execute test results 212a can list whether or not the test 208a executed correctly, and if not, the reason for the execution failure. The test results 214a-214b list the actual test output for the particular test. For example, test result 214a can list the actual test output for test 208a, including whether the test passed or failed, and in the case of failure, why the failure occurred and where the failure occurred. To automate the test cycle, embodiments of the present invention define [[a]] an XML document type definition (DTD) for the test result phase 214.

Page 11, lines 4 to 15:

The results of the test execution are then captured in a test execution log file 200 which includes detailed descriptions of the tests that were executed and the results of each test, as described above with respect to Figures 2A and 2B. To generate the test results included in the test execution log file 200, embodiments of the present invention make function calls to a status utility 406. The status utility 406 includes functions that generate XML statements in accordance with the test DTD 300, discussed above with respect to Figure 3. In particular, the test application 402 includes function calls to the functions provided in the status utility 406. These ~~function~~ functions return XML statements in accordance with the test DTD 300, which are then written to the test execution log file 200. As a result, the test execution log file 200 includes test results that are XML enabled, in addition to non-XML enabled compiler and execution information as described above with reference to Figure 2B.

Page 12, lines 12 to 22:

Figure 5 is a logical diagram showing a process cycle 500 for generating a test summary report, in accordance with an embodiment of the present invention. As shown in Figure 5, the test execution log file 200 is input to a parser 502, which processes the test execution log file 200 to produce a well-formed XML test report file 504. It should be noted that the parser 502 can also be implemented as a Java utility using a Java code. As mentioned above, the test execution log file 200 includes information other than test result information. For example, the test execution log file 200 often includes compiling and execution information that is used in debugging and other test maintenance operations. The parser 502 processes this ~~information~~ extra information to create the well-formed XML test report file

504. In addition, the parser 502 can extract control characters not utilized during further operations of the process.

Page 13, lines 8 to 21:

As mentioned above, the parser 502 creates a well-formed XML test report file 504. In addition, embodiments of the present invention create the well-formed XML test report file 504 such that the XML enabled test reports are valid as well, according to the Test DTD. A well-formed XML document is [[a]] <u>an</u> XML document that complies with XML well-formedness constraints. These constraints require that elements, which are named content containers, properly nest within each other and use other markup syntax correctly. Unlike HTML, well-formed XML elements are defined by their use, not by a rigid structural definition, allowing authors to create elements in response to their development. A valid XML document is [[a]] <u>an</u> XML document that conforms with a corresponding DTD. As mentioned above, [[A]] <u>a</u> DTD is a set of rules that a document follows, which software may need to read before processing and displaying a document. These rules generally state the name and contents of each element and in which contexts it can exist. Paragraph elements might be defined as containing keyword and code elements and as existing within section and note elements.

Page 14, lines 4-23 to page 15, lines 1-5:

As each test is executed, the results are written to the test execution log file 200 using the status utility, as described above with reference to Figure 4. However, during a particular test cycle, a test engineer may run any number of tests that are available in a particular test suite. That is, although a particular test suite may include a predefined number of tests, the test engineer may run all, some, or none of the tests within the particular test suite. Thus, the actual number of test executed in a particular test suite may not be known until the test suite is actually executed. The embodiments of the present invention address this issue by writing test results to the test execution log file 200 in an independent manner. Specifically, each test result includes information identifying the test and the test suite to which the test belongs. Hence, the test results of the well-formed XML test reports file 504 are arranged as a plurality of independent test results, each identifying its test ID and the test suite to which the test belongs. For example, using the exemplary DTD 300 of Figure 3, the Suite-Name attribute 310 can be used to logically arrange the tests. In this embodiment, all ~~test~~ tests belonging to the same test suite, as indicated by the Suite-Name attribute 310, are arranged together under one XML element.

In one embodiment, the logical XSLT stylesheet parser 506 is written using XSLT. XSLT is a language used to convert [[a]] an XML document into another XML document or into HTML, PDF, or some other format. The conversion is accomplished with a XSLT processor, which transforms the input based on XSLT extensions of the XSL stylesheet. XSL statements are also followed. The processor uses [[a]] an XML parser to separate the XML elements into a tree structure, which the processor manipulates. Although, a logical XSLT stylesheet parser 506 is illustrated in Figure 5, it should be noted that the logical parser 506 of the embodiments of present invention can be developed utilizing any computer programming language, such as Java, C, Assembly, or other computer programming languages as will be apparent to those skilled in the art.

Page 15, lines 14-22 to page 16, lines 1-5:

Once the test results are logically arranged within the logically arranged XML test report file 508, ~~a~~ an HTML converter XSLT stylesheet parser 510 converts the logically arranged XML test report file 508 into ~~a~~ an HTML test summary report 512. As the name implies, the HTML test summary report 512 comprises HTML code, which can be interpreted using a browser. HTML is the set of markup symbols or codes inserted in a file intended for display on a browser. The markup symbols are a sequence of characters or other symbols that are inserted at certain places in a text or word processing file to indicate how the file should look when it is printed or displayed or to describe the document's logical structure. These markup indicators are often called "tags."

The markup tells the browser how to display ~~a~~ an HTML page's words and images for the user. Each individual markup code is referred to as a "tag." Some tags come in pairs that indicate when a particular display effect is to begin and when it is to end. HTML is generally adhered to by the major browsers, Microsoft's Internet Explorer and Netscape's Navigator, which also provide some additional non-standard codes.

Page 16, lines 15-23 to page 17, lines 1-4:

In addition, since the HTML test summary report 512 is written in HTML, a link 515 can be provided for test failures. The link 515 provides access to other HTML pages that describe the failure and why the failure occurred. In the case of failures, embodiments of the present invention can use the test execution log file 200 to determine where the failures are occurring and why the failures are occurring. These results can then be summarized in the HTML test summary report 512 and accompanying failure description pages 516. For example, when a user selects a link 515 for the failures of test suite X, the user is presented with a failure description page 516 describing the test failures of test suit X and why the failures occurred. The HTML test summary report 512 can then be distributed to the appropriate personal, such as the project manager or development team. Embodiments of the present invention can also describe test failures within the same document as the test summary report 512 and use local links to access the failure descriptions.

Page 19, lines 3 to 11:

Once the test results are logically arranged within the logically arranged XML test report file, [[a]] an HTML converter XSLT stylesheet parser converts the logically arranged XML test report file into [[a]] an HTML test summary report, in operation 610. The HTML test summary report comprises HTML code, which can be interpreted using a browser. The HTML test summary report can be displayed on a browser, thus, presenting the user with a summary of the detailed testing information included in the test execution log file. In addition, the HTML test summary report can include a summary analysis of the test execution log file. Specifically, the HTML test summary report can list the tests that were executed and whether the tests passed or failed.